XGBoost: A Scalable Tree Boosting System

Published Year: 2016
Author: Chen Tianqi

TREE BOOSTING IN A NUTSHELL

minimize the following regularized objective

$\mathcal{L}(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right)$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$
T: the number of leaves
w: leaf weights
Ω penalizes the complexity of the model

- a continuous score on each of the leaf, we use wi to represent score on i-th leaf.
- the regularized objective will tend to select a model employing simple and predictive functions.

Shrinkage

- Shrinkage scales newly added weights by a factor η after each step of tree boosting.
- Similar to a learning rate in tochastic optimization, shrinkage reduces the influence of each individual tree and leaves space for future trees to improve the model.

Column Subsampling

- column sub-samples also speeds up computations of the parallel algorithm described later.
- subsampling columns not only reduces running time, and but also gives a bit higher performance for this problem. This could due to the fact that the subsampling helps prevent overfitting, which is observed by many of the users.

Gradient Tree Boosting $\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(\mathbf{x}_i\right)\right) + \Omega\left(f_t\right)$
$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right)$ Second-order approximation can be used to quickly optimize the objective in the general setting.

should know more about: taylor expansion

- https://en.wikipedia.org/wiki/Taylor_series
- https://en.wikipedia.org/wiki/Taylor's_theorem

$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$ The formular can be used as a scoring function to measure the quality of a tree structure q. This score is like the impurity score for evaluating decision trees, except that it is derived for a wider range of objective functions.

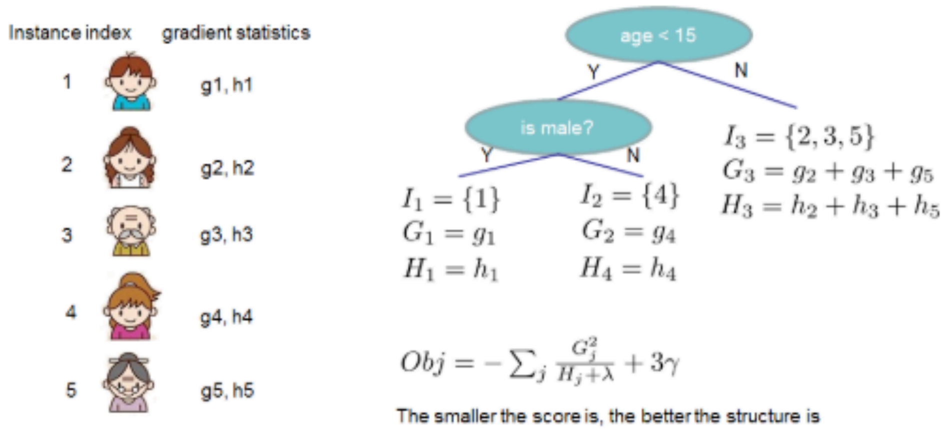


Figure 2: Structure Score Calculation. We only need to sum up the gradient and second order gradient statistics on each leaf, then apply the scoring formula to get the quality score.

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i+\lambda}\right] - \gamma$$

This formula is usually used in practice for evaluating the split candidates.

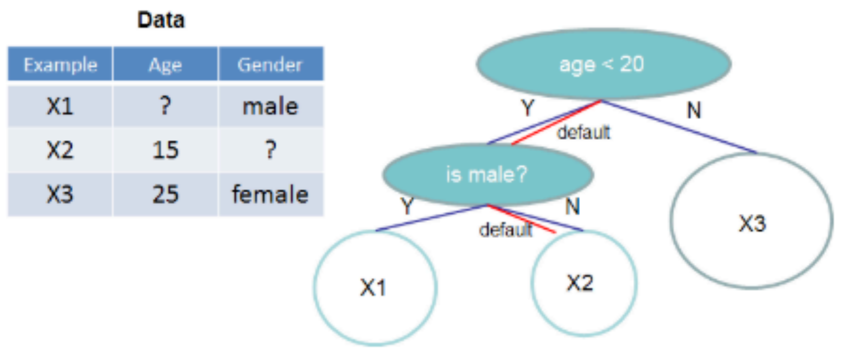
Find split points

The local proposal refines the candidates after splits, and can potentially be more appropriate for deeper trees. We find that the local proposal indeed requires fewer candidates. The global proposal can be as accurate as the local one given enough candidates.

Approximate Algorithm

To summarize, the algorithm first proposes candidate splitting points according to percentiles of feature distribution (a specific criteria will be given in Sec. 3.3). The algorithm then maps the continuous features into buck- ets split by these candidate points, aggregates the statistics and finds the best solution among proposals based on the aggregated statistics.

Handle missing values


Data

Example	Age	Gender
X1	?	male
X2	15	?
X3	25	female

Figure 4: Tree structure with default directions. An example will be classified into the default direction when the feature needed for the split is missing.